NonStop® Advanced Architecture

David Bernick, Bill Bruckert, Paul Del Vigna, David Garcia,
Robert Jardine, Jim Klecka, Jim Smullen
Hewlett Packard Company
{David.Bernick, Bill.Bruckert, Paul.DelVigna, Dave.Garcia, Robert.Jardine, Jim.Klecka, Jim.Smullen}@hp.com

Abstract

For nearly 30 years the Hewlett Packard NonStop Enterprise Division (formerly Tandem Computers Inc.) has produced highly available, fault-tolerant, massively parallel NonStop computer systems. These vertically integrated systems use a proprietary operating system and specialized hardware for detecting, isolating, and recovering from faults. The NonStop Advanced Architecture (NSAA) uses dual or triple modular redundant fault-tolerant servers built from standard HP 4-way SMP Itanium®2 server processor modules, memory boards, and power infrastructure. A unique synchronization mechanism allows fully compared operations from loosely synchronized processor modules. In addition, the NSAA improves system availability by additional hardware fault masking, and significantly lowers cost by leveraging existing high-volume Itanium server components.

1. Introduction

The NonStop system is a massively parallel cluster of independent processors each running its own copy of the operating system [1]. The processors communicate with each other and with shared I/O adapters through the ServerNet® [2] system area network (SAN). A single hardware failure can disrupt at most one processor, I/O adapter, or interconnect path. Even with a failure, the redundancy in the system allows the user's application to continue uninterrupted.

NonStop systems rely on self-checked processor modules that provide a simple guarantee: they either provide the correct result, or they promptly stop and emit no result, preventing incorrect data from propagating elsewhere in the system.

The methods used to achieve self-checked processors have evolved over the span of different NonStop processor products. Early systems used custom designed processors with self-checking techniques such as redundancy codes to detect failures. For the last sixteen years, all NonStop processors have been built with tightly lock-stepped microprocessors. Two microprocessors using the same clock have their outputs compared after each operation and immediately signal any discrepancy. See figure 1. Redundancy codes in the memory and cache ensure that any memory errors are corrected or result in the immediate stopping of the processor.

Future NonStop systems will not be able to use these same techniques. Trends in microprocessor design mean it is no longer viable to duplicate and compare tightly lock-stepped microprocessors. For example:

- Minor nondeterministic behavior, such as an arbiter of asynchronous events, will not affect normal operation but will disrupt lock-stepped operation of the microprocessors.
- Power management techniques with variable clock frequencies cannot be used with lock-stepped microprocessors.
- Multiple very high-speed functional blocks integrated onto one die result in multiple clocks and asynchronous interfaces, making lock-stepping difficult (or impractical).
- Smaller die geometries result in higher soft-error rates and require low level fix-up routines. This greatly complicates lock-stepped operation of the microprocessors.
- Future microprocessors will predominantly be Chip Multi-processors (CMP) and have multiple processor cores on each die. NonStop systems rely on the fact that a single failure can disrupt at

most one processor in the system. A failure in a CMP would disrupt multiple processors – something the pre-NSAA NonStop system architecture cannot tolerate.

In addition, market forces produce continual pressure to reduce the cost of product and cost of development. High volume 4- or 8-way SMP servers have significantly better cost/performance than previous high-data-integrity NonStop hardware. Leveraging these servers would be desirable, but there are significant problems, not the least of which is that these servers are not self-checking.

The NonStop Advanced Architecture (NSAA) is a new way of building fully self-checked processors without requiring lock-stepped comparison of microprocessors. NSAA systems can deterministically run an application on multiple microprocessors without requiring custom processor boards running the exact same operations on each clock cycle. NSAA uses slightly modified traditional servers in a dual or triple modular redundant configuration. The redundant processors are said to be in loose lockstep. That is, they deterministically run the same application instruction stream, but they are allowed to run at different clock rates, to independently do error retries or fixup routines, and to hit or miss cache(s) at different points.

This paper begins with a description of the NonStop architecture and explains the methods for achieving high reliability. Next the new NonStop Advanced Architecture is explained. Further sections describe key features of the NSAA including voting, the synchronization of IO and interrupts, as well as online replacement of a processor module.

2. NonStop System Background

NonStop systems are used in applications that require the very highest levels of availability, data protection, or scalability, such as automatic teller systems, credit card authorization, retail point-of-sale, stock trading, funds transfer, cellular phone tracking and billing, 911 emergency calls, electronic medical records, travel and hotel reservations, and electronic mail.

Today, well-managed NonStop systems achieve "five nines" (99.999%) application availability in the face of unplanned outages (based on customer-reported outages attributable to NonStop hardware, software, or processes). The goals for data integrity protection are similarly ambitious: these NonStop systems are expected to experience undetected data corruption at a rate of less than one FIT per processor (one undetected error per billion hours of operation). This data integrity goal is two or more orders of magnitude smaller than expected for an unchecked microprocessor [3].

With respect to scalability, our goal is to approach 100% linear scalability from a few to thousands of processors. In this context, scalability means that the amount of incremental useful work done when any processor is added to a cluster is essentially the same as the incremental amount of work that was accomplished when each earlier processor was added. This goal contrasts with conventional systems, for which scalability degrades significantly as a system or cluster grows from 8 to 16 to 32 to 64 processors, etc. NonStop systems are not subject to shared-memory multiprocessor scaling issues such as memory bandwidth contention, shared lock contention, and cache-line contention.

NonStop systems comprise multiple processors, I/O adapters, storage devices, communications lines, and system area networks (SAN) (figure 1). Two or more of each of these components are used in an all-active, 'N+1' configuration, providing full backup in the event of failure. Each processor is associated with its own memory; there is no memory shared among processors. Instead, the processors are independent and communicate with each other only via messages passed over the SAN. The SAN connects all of the processors to each other as well as to all I/O adapters. Because all system components (processors, I/O adapters, I/O devices, and the SAN itself) are replicated, no single fault can stop the user's application.

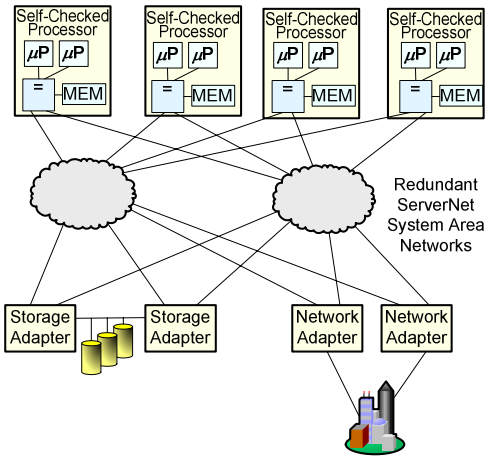

Figure 1: 4 Processor NonStop System with Duplicated and Compared Microprocessors

Each of these hardware components is self-checking and (except for simple correctable or retryable errors, such as many memory errors and I/O

timeouts) implements a fail-fast philosophy: when an error is detected, the component stops. It either works correctly or removes itself from the system. This approach provides a high level of fault containment and also serves to unambiguously identify the failing component. When a component fails for any reason, its workload is distributed among the remaining components, thus providing fault tolerance.

The operating system executing on these systems is the NonStop Kernel [4]. It provides the usual operating system features (memory management, process control, etc.) along with a sophisticated message system and a variety of mechanisms to recover from failures, both hardware and software.

Critical system software is implemented as process pairs, a primary and backup copy of the same program. The primary process performs the actual work of the pair and communicates state changes to the backup process via interprocessor messages. When a fault occurs, the backup process can take over the responsibilities of the pair. Such a fault could be either a hardware fault (e.g., the failure of a processor) or a software fault (e.g., the failure of the primary process due to a consistency check, an invalid address, or similar fault). When a backup process takes over from its primary process, the message system automatically routes messages intended for that process pair to the former backup (now primary) process.

(Process pair takeovers due to failure of the primary process happen immediately. Takeovers after a processor failure, whether done because of a hardware or a software fault, are usually accomplished within a few seconds. This time is long enough to prevent the NonStop architecture from serving extreme real-time requirements, but it is quite satisfactory for commercial on-line transaction processing applications, and it is very short when compared with so-called "high-availability" clustered solutions.)

Twenty-five years ago, application writers had to write process pairs by hand and undertake the difficult job of assuring that the backup could recover in all cases. Modern applications avoid this work by using system-supplied middleware such as the NonStop Pathway [4] or Tuxedo transaction monitor. By layering on top of this middleware, the user's application becomes fault tolerant with no additional work or specialized knowledge required.

With N+1 processor redundancy, the workload of a failing processor is shifted to the remaining processors without any disruption to the user's application. A single processor failure is not a critical problem, as it does not result in a customer system outage. After a processor is replaced, the workload migrates back to the new processor. (Workload shifting in transactional OLTP applications is done by simply routing new transactions to server processes in the other processors.)

The system area network, ServerNet, is a high-speed, low-latency, packet-switched network, used for both inter-processor communications (IPC) and input-output (I/O). It comprises two independent fabrics, connecting all processors to each other as well as to all I/O adapters. A single failure in ServerNet can disrupt at most one fabric. Packets are protected by a (CRC) checksum; lost or corrupted packets are retransmitted, first on the same fabric, and then, if necessary, on the other fabric. Both fabrics are typically used at all times by the various processors making independent choices of the fabric to be used for any given message.

ServerNet provides "memory semantics" similar to InfiniBand® [5] or RDMA/Ethernet [6] (although ServerNet does not provide direct user-mode access to RDMA). Data can be pushed (remote write) or pulled (remote read) between any pair of components on the SAN, with appropriate permissions established by the owner of the resource (which can be enforced at byte granularity). Thus, for IPC, the message system uses the SAN to send and receive messages to/from other processors. For I/O, the I/O adapters use the SAN to pace the movement of I/O data to and from the host (processor) memory.

Fault tolerance for storage data is provided by two mechanisms: end-to-end checksums for error detection and mirrored volumes for data recovery.

Logical disk volumes are implemented by a pair of mirrored drives or Logical Units. Writes are performed to both volumes; reads are optimized to use the copy that should provide the quickest access time. In case of a read error, the data can be read from the mirror copy. In case of a catastrophic media failure of one of the drives in a logical disk volume, the drive can be replaced and "revived" with the contents of the other disk drive in an on-line operation. Optional Enterprise Storage Systems can also mask many disk failures from both application software and even the operating system.

Disk data is protected by end-to-end checksums. For each block, a checksum covering the data and block address is calculated in the processor, written to disk along with the data, and verified by the processor when the block is read. In the event of a checksum error, the data is read from the other member of the mirrored pair.

Layered on top of the NonStop Kernel are a

distributed, transactional, relational database (NonStop SQL), and other transaction processing software. These software layers are designed and tightly integrated with the operating system to provide high performance, scalability, and high levels of fault tolerance and data integrity protection. In addition, a distributed file system and middleware present an integrated single-system image to application software and to external (LAN and WAN) clients.

3. The NonStop Advanced Architecture

The new architecture retains the overall logical structure and all of the features of the previous NonStop architecture, but it uses a different method of error detection in the processors.

Instead of lockstepping microprocessors, the NSAA detects processor failures by comparing the outputs of I/O operations (both IPC and device I/O) from two or three slightly modified high-volume 4- or 8-way SMP servers. The two or three servers execute similar instruction streams, each running on an independent clock. All outputs from the servers are compared for 100% detection of faults. Because the primary point of comparison is on I/O output operations, variations in operation for error handling, e.g. error correction, cache retries etc. are tolerated and do not result in a processor miscompare, unlike in traditional lockstep mechanisms.

Either dual or triple modular redundant (DMR, TMR) configurations of Itanium servers are used in the NSAA. Each server is slightly modified to support reintegration (see Reintegration below) and output comparison (see Voter below). Both DMR and TMR provide full detection of faults. While DMR is more cost effective, TMR is capable of unambiguous determination of which server is in error and allows uninterrupted operation, even after a failure.

DMR meets the requirements for fault tolerance in a NonStop System: it provides unassailable detection and isolation of failures within a processor. Since the NonStop System can tolerate the stopping of a single processor, DMR adequately ensures continuous application availability for the customer.

A successful fault-tolerant computer minimizes the impact of hardware and software errors on the customer's application. However, that improvement means that human errors, such as erroneous service actions, become a more significant cause of outages. For example, a user attempting to replace the lone working module of a redundant pair can cause the application to fail. The NSAA TMR system is resilient to such faults. Removing the wrong module in a TMR system reduces the degree of redundancy but does not stop processor operation.

Another benefit of TMR is that it shields old applications that were neither written as process pairs nor layered above fault-tolerant middleware from hardware failures. Such non-fault-tolerant applications would be disrupted by a single processor outage. TMR removes almost all single hardware causes of failure.

In a TMR NonStop system, three 4-way SMP servers operate as four processors of the scalable NonStop system. Given this arrangement, the term "processor" is ambiguous. Instead, the NSAA defines the terms 'slice', 'logical processor', and 'processor element' (PE). Figure 2 shows the three 4-way SMP servers as columns, each referred to in the NSAA as a slice. Each row in figure 2 represents a logical processor. In a NonStop system, the logical processor is the self-checked member of the cluster. The logical processor is made up of processor elements (PEs), one from each of the slices. Each processor element is a microprocessor running its own instruction stream and has a portion of the slice memory dedicated to its use. There are no synchronized clocks among the slices. Each PE runs asynchronously of the other PEs in the logical processor.

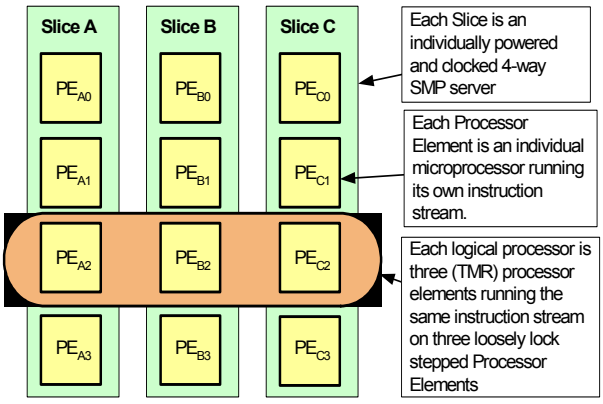

Figure 2: Four Logical NSK Processors built from TMR 4-way SMP servers.

Processors in the NonStop architecture do not share memory among themselves; instead, each maintains its own copy of the operating system and communicates only through the system area network (SAN). Even though the NSAA hardware supports shared memory, it is not used by the NSAA processors. The memory system in each slice is partitioned so that each PE can access only its own portion of the memory. This isolation of each PE's memory space is enforced through unique virtual to physical mappings as well as the Itanium protection

key mechanism [7].

All I/O from the logical processor goes through the system area network, which in turn connects to additional TMR/DMR processors and to I/O adapters. There are no individual I/O adapters on the slices. The I/O section of each standard high-volume server is replaced with serial links tying the two or three server boxes to 'logical synchronization units' or LSUs (figure 3). The fully self-checked LSU contains the voting logic and SAN interface for one logical processor. The voting unit compares all output operations of a logical processor, ensuring that all slices agree on a result before the data is written to the system area network. Should one of the slices disagree, the voting unit selects the data from the other two and software stops the failing PE.

The NSAA allows each logical processor to have either one or two LSUs. If a logical processor is configured with a single LSU, an LSU failure stops the logical processor. Such a failure does not stop the customer's fault-tolerant application because LSU replacement does not affect the slices (it is an independently-replaceable unit), and the NonStop system can tolerate a single logical processor failure.

For those customers seeking to be shielded from any possible hardware induced failure, a second LSU can be associated with the logical processor. The second LSU enables four independent SAN fabrics. With two LSUs per TMR logical processor, the system can be configured to be fully tolerant of any two hardware faults.

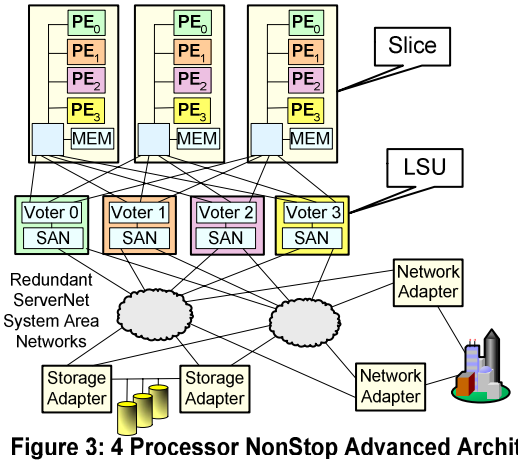

Figure 3: 4 Processor NonStop Advanced Architecture System TMR Configuration

3.1 Symmetric Memory State

When fault-tolerant microprocessors are run on the same clock they are referred to as lock-stepped or tightly synchronized. By contrast the NSAA is loosely synchronized. Each PE of the logical processor updates memory so that on any output operation the data pulled from each slice's memory is the same. To keep the memories symmetric, each PE executes basically the same instruction stream.

The PEs of a logical processor all have the same virtual to physical mapping and all take page faults at the same point in the instruction stream. However, they need not all have the same cache state and TLB entries. One PE can miss in cache or TLB while the others hit. This flexibility is allowed since the only requirement is that I/O operations of each PE be the same.

The requirement that the PEs do symmetric page faults limits the NSAA's ability to exploit Itanium data and control speculation [7]. Even though the speculative code executed by an Itanium processor is not necessarily symmetric across all PEs, the resulting state of memory is deterministic. Unfortunately, the speculation fixup routine that might execute on one PE and not the other could create an asymmetric page fault. Lacking the compiler restrictions to prevent these asymmetric page faults, the NSAA takes the safe path and disallows all data and control speculation that uses independent speculation fixup routines. The NSAA allows only the inline form of Itanium data speculation fixup. Single instruction check loads, (e.g. ld.c) are allowed, but to avoid asymmetric page faults the NSAA does not allow advanced load check fixup routines (e.g. a fixup routine reached by a chk.a instruction) or any form of control speculation.

At any point in time each PE will be slightly ahead or behind the other PEs in the logical processor. The process scheduling and interrupt handling algorithms on each server are modified so that asynchronous inputs are acted upon by each server at the same point in their instruction stream (See Rendezvous below).

Input data from the system area network is written into each slice's memory. Incoming packets from the system area network arrive in each PE's memory at slightly different times. This data is considered asymmetric until an I/O completion notification informs all the PEs that the data has arrived. A program must not read incoming data before the completion notification. To prevent an application program from reading this potentially asymmetric data, the operating system marks the page inaccessible until the I/O operation completes. If an application reads or writes the page prematurely, a fault is generated and the process is made inactive until the I/O operation completes. Similarly, a process is not allowed to modify an active outbound I/O buffer.

To ensure symmetric operation, the time-of-day (TOD) value used in the logical processor must be synchronized for each PE. On a typical system, the TOD is derived directly from a timer in the microprocessor. Such a timer is unusable in the NSAA since each slice executes instructions at a slightly different rate due to asymmetric cache misses and slightly different clock oscillators. To solve this problem the LSU hardware provides a TOD value for the entire logical processor. Software can access this TOD value by reading a register in the LSU. For higher performance, the TOD value either can be pushed to each PE during the Rendezvous operation (see below) or can be deterministically estimated by each PE.

3.2 Logical Synchronization Unit: Voter & SAN Interface

The Logical Synchronization Unit (figure 3) is NSAA specific hardware that provides the voter and SAN interface function for the processor. There are one or two LSUs per logical processor. For an NSAA system built from 4-way SMP slices, there would be four or eight LSUs for the four logical processors. Failure of any one LSU affects only a single logical processor. The LSU (both the voter and SAN interface function) is designed with a completely self-checking methodology [8] to ensure that any data passing through it is not corrupted. The LSU is designed to either function correctly or self-detect the failure and shut down, thus isolating failures from the rest of the system. An LSU can be replaced on-line without affecting any of the other logical processors. The failure rate of an LSU is expected to be a small fraction of the failure rate of a PE. For the first implementation of the NSAA, the LSU is built from an FPGA voter and an ASIC ServerNet SAN interface.

The voter logic ensures that any data leaving the logical processor is agreed to by a strict majority of the slices. The voter logic checks all PIO (Programmed I/O) loads and stores from the PEs to ensure that the identical operation is coming from each slice. Since all I/O operations from the logical processor go through the SAN interface, the voter guarantees that any data going onto the SAN is agreed to by all the slices. The voter logic not only keeps any failure from propagating outside the logical processor, but also can often identify which slice is in error.

In a TMR configuration, a voter miscompare unambiguously identifies which slice is at fault. With the failing slice identified, the I/O is allowed to complete with the good data. Application software continues running on the logical processor, and low level system software can shut down the failed PE. After a failure, the PE can either be restarted and reintegrated or the entire slice can be replaced. Restarting the PE is the best choice for a random soft error. If the error repeats, replacement is in order. A three-way voter miscompare results in halting of the logical processor, although there is no known single failure mode that could cause such a miscompare.

In a DMR configuration, a voter miscompare can be ambiguous as to which slice is at fault. Certain errors, like a bus fault, are self-identifying and allow continued operation on the good slice. But if the voter logic simply detects a miscompare with no other information, the logical processor must be halted. Since the NonStop architecture can tolerate the loss of a single logical processor, the application will fail-over to other logical processors and continue to run.

The NSAA offers an option to add a simple heuristic that may disambiguate a DMR voter miscompare. The heuristic uses a probation vector with one bit for each possible slice. If a bit is set and there is an ambiguous DMR vote miscompare, the voting is resolved either in favor of the slice that does not have its probation bit set or against the slice that is unable to access the probation bit. The probation bit is set for a short time after a slice with a known error is restarted or if the slice has been experiencing excessive correctable errors. On the assumption that a slice already identified as having problems will tend to exhibit more problems, the bit allows hardware to choose which slice's I/O to block and which to allow. Probation bits are expected to be only infrequently set and then only for a short period.

3.3 Reintegration

Reintegration is the process where Processing Elements (PEs) on a single slice are brought back into loose synchronization with PEs on other slices. Reintegration is required after a new slice is installed or replaced or after a voting error results in a stopped PE. After just a single error, a failing PE might be reintegrated with the rest of the logical processor on the assumption the voting error was the result of a transient error and not a persistent fault.

The reintegration procedure copies the state of the running, on-line PE's memory into the memory of the newly added PE. After the memory state is copied, the new PE starts executing the exact same instruction stream as the other PEs. The reintegration operation happens while the logical processor is online and executing the customer's application. All memory writes that occur in the source PE are also copied to

the target, assuring a consistent memory image. Reintegration consumes only a small percentage of CPU utilization and takes less than ten minutes to complete for a 32 GByte logical processor.

To facilitate reintegration, the standard 4- or 8-way SMP server is modified to support a reintegration link (figure 4). The reintegration link forms a directional ring connecting the two or three slices. The reintegration link connects to logic inserted between the processor chipset and the main memory. The logic replicates each local write to memory and sends it across the reintegration link to the neighbor slice. A slice in normal operation ignores the input from the reintegration link. But during reintegration, the reintegration target feeds its memory with write operations from the reintegration link. The reintegration link for our Itanium2 servers supports full-speed memory writes transmitting 7.5 GByte/s over 24 optical fibers, each operating at 3.125 Gbit/s. The reintegration links are CRC protected, and correct link operation is checked even when the links are not in use for a reintegration operation.

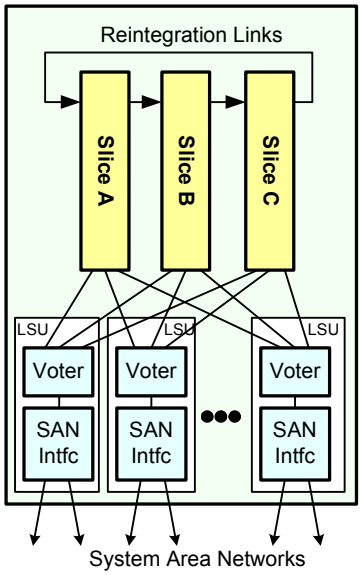

Figure 4: Reintegration Link Reflects Memory Writes in one Slice to the Next Slice

The reintegration procedure consists of several steps:

1) The reintegration target PE is quiesced and executes a tight, in-cache loop, not accessing memory.

2) The reintegration target PE switches its memory system to accept write operations from the incoming reintegration link.

3) A background reintegration process is started on the running processor elements. The process sweeps all of physical memory forcing a write-back of each cache line (by touching each cache line with an atomic read and write of the same data). This causes all of the memory contents of the source slice to be copied to the reintegration target. DMA write operations from the SAN as well as memory writes from the active processes are also copied to the reintegration target.

4) At the end of the sweep, the running PEs are momentarily stopped while their internal state is written to memory and their caches are flushed to memory. After the flush, the reintegration target memory is identical to that of the source.

After the flush completes, all the processor elements (including the reintegration target) can resume operation. All the PEs are executing the same instruction stream and memory is symmetric among the PEs of the logical processor. Except for the brief (a few milliseconds) disruption in step (4), the user's application continues to run throughout the reintegration operation. This disruption is much less than the disruption caused by a processor takeover in the traditional NonStop architecture.

3.4 Rendezvous

Interrupts to the processor, such as I/O completion interrupts, arrive at each PE at slightly different times. If the processor were to immediately handle an interrupt, each PE would be interrupted at a different point in its instruction stream. This divergent execution would result in asymmetric memory state in the logical processor. In order to keep memory symmetric, each processor element must handle an interrupt at the exact same point in the instruction stream as the other PEs in the logical processor.

To do symmetric interrupt handling, the NSAA defines a scheme for synchronizing the execution of interrupt handler code. This "rendezvous" scheme consists of:

- Hardware in the Voter logic that writes up to 32 bytes of data from each PE to the rest of the PEs in the logical processor.
- A small section of code called a Voluntary Rendezvous Opportunity (VRO) embedded throughout the OS and implicitly invoked by user applications.
- A rendezvous protocol, in which each PE proposes a certain VRO that, when reached, will schedule the interrupt handler for execution.

After receiving an interrupt, each PE initiates a rendezvous operation. A rendezvous operation consists

of each PE writing special rendezvous registers in the voter logic. After all the writes complete, the voter reflects the data from the rendezvous registers back to a specific block of memory in each PE. Rendezvous code in each PE then reads the block of data to see what the other PEs wrote.

The PEs use the block of data to propose where in the instruction stream they intend to execute the interrupt handler code for a specific interrupt. The proposal consists of an interrupt identification and a VRO sequence number in the near future. Each PE compares its proposal with that of the other PEs and selects the highest proposed VRO sequence number. When each PE reaches that VRO it can symmetrically schedule the interrupt handler for execution.

The VRO is a small section of code inserted into the code stream. At a minimum, the VRO code will be inserted into every privilege level transition. The highly optimized VRO code takes only a handful of instructions to execute in the typical case where there is no synchronization action required. The VRO code increments an identifying sequence number and checks to see if any interrupt handlers are pending execution at this particular VRO.

The intent is to execute VRO code periodically. The VRO serves as a landmark that identifies a particular point in the symmetric instruction stream that is the same in each of the PEs in the logical processor. This landmark can then be used to process interrupts or dispatch new processes all the while maintaining symmetric memory state among the PEs of the logical processor.

3.5 Uncooperative Processes

When a process executes for a long time without encountering a VRO, it is referred to as "uncooperative" and must be interrupted. Uncooperative processes are undesirable in that they increase the latency to handle interrupts and delay the dispatching of higher priority processes. Due to automatic embedding of VROs, the typical transaction processing workload run on NonStop systems is expected to be cooperative.

The interval timer interrupt used to detect long inter-VRO intervals does not occur at the same point in the instruction stream of each processor element of the logical processor. In order to ensure a symmetric memory state, each PE must be brought to the same point in its execution before the uncooperative process can be suspended and another process dispatched. The NSAA can use any one of several solutions to do this.

For our Itanium-based implementation, pure instruction counting based on the retired user-level instruction count alone may not be sufficient to synchronize the PEs, as its function is to monitor performance, not to count perfectly in every circumstance. So, we don't trust it to be perfect -- we assume only that its error is bounded and reasonably small over a short time interval. After first synchronizing based on the count of retired user-level instructions, we assume all PEs are within a bounded number of (actual) instructions, N. By counting only user-level instructions we allow low-level system code to do soft-error fixups on one PE without affecting the instruction count.

We define two algorithms to bring the PEs to the exact same point in the instruction stream, called UNCP-Store and UNCP-Trace.

The UNCP-Store algorithm identifies process state that could be different in the PEs. It then chooses one PE (the source PE) and copies its values for that state to the other (target) PEs, putting the target PEs at the same execution point as the source PE.

To identify process state that differs among the PEs, breakpoints are set in each PE so that all user stores are trapped. Each PE is executed for at least N instructions, trapping on every store. The trap handler simply saves the address that is being stored and resumes the process. After this step each PE has a set of locations they have modified. The union of these sets represents all the memory locations that may differ. One PE is chosen, and its values for those locations are copied to the other PEs. Finally, the register state is copied from the chosen PE to the other PEs. All copies of the uncooperative process are now synchronized.

Due to copying some state from the source to the target PE(s), the UNCP-Store algorithm is vulnerable to error propagation. We expect this risk to be quite small, both because the algorithm should be executed infrequently and the amount of state that is copied should be small. The UNCP-Trace algorithm, described next, addresses this vulnerability.

The second algorithm, UNCP-Trace, does not copy state from one PE to the others. Instead, it determines which PE is ahead, and by how far, and executes instructions in the trailing PEs until they are synchronized with the leader.

Once the PEs are all at the same retired instruction count, each PE does single-step execution for N to 3N instructions. As a PE executes the instructions, it records the instruction pointer (IP) and the inputs of each instruction. For an add instruction at IP 200, with addends 5 and 9, it would record 200:5,9. The PEs exchange and correlate their instruction traces to

determine where they are relative to one another. The matching algorithm (not described here) determines the length of trace necessary to determine the leader, which PE is the leader, and how far the trailing PEs must execute to catch up.

Once the PEs are synchronized, by either of these methods, one final step is taken. A VRO is inserted into the code at the point of synchronization. This change prevents the process from becoming uncooperative again, at least at this point in its execution.

3.6 Asymmetric Memory Dump

The unique capabilities of the NSAA provide additional benefits. As an example, they are exploited in a method that allows for a post-failure dump of processor memory while still immediately reloading the logical processor. In previous systems, preserving the memory state of a failed processor for later analysis of software faults meant delaying the return of that processor to full availability. With redundant processor and memory state available in the NSAA, one of the dual or triple redundant processor/memories is preserved for the memory dump while the other one or two processor/memories are immediately reinitialized to run the operating system. After the memory dump completes, a reintegration operation loads the "preserved" processor element with the currently running state of the other one or two processor elements, thus returning the logical processor to its fully redundant state.

4. Performance Implications of the NSAA

NSAA is designed to offer the highest levels of availability required by the most demanding customers. How much does this architecture affect performance in comparison to conventional uniprocessors? At this stage in the design, some of the answers are estimates, but confidence is high that the performance implications of NSAA are small.

- I/O latency is slightly increased due to voting the results; we expect this impact to be negligible;
- Memory latency is increased due to the memory copy hardware, which is in the path to memory even when reintegration operations are not active; based on modeling of processor caches and the memory subsystem, we expect this impact to be small, but not negligible, probably in the 5% percent range;
- Interrupt-handling latency is slightly increased due to the voluntary rendezvous mechanism of handling interrupts at the same logical time in each processor element; in most on-line applications, we expect this extra latency to have a very small impact;
- We do not currently take advantage of the Itanium speculation features. At least for current Itanium processors, we expect only a few percent performance impact for most applications. Because future Itanium speculation features may be more effective, we are planning additional work to allow the use of these features in future generations of NSAA processors.

In summary, we expect the negative impact of NSAA to be in the range of 5% to 10% compared to a more traditional uniprocessor approach. However, balancing those performance decreases are several significant benefits that NSAA provides relative to our previous designs:

- NSAA allows increased clock frequency relative to a more traditional tightly-lockstepped processor; this may become an even larger benefit with the advent of power management techniques with variable clock frequencies;
- NSAA allows NonStop processors to make use of cost effective chip multi-processors. The traditional NonStop architecture cannot otherwise run on SMP hardware;
- NSAA allows the development of processors with significantly lower development costs, which is a significant benefit given the relatively low volumes in this market segment.

5. Competitive Approaches

Prior to the mid-1980's, fault-tolerant computers for enterprise class data processing used custom-designed processors with redundancy techniques such as parity predicting ALUs to ensure correct operation of the processor. With the advent of microprocessors, these techniques were dropped in favor of duplicating and comparing standard microprocessor parts. Standard microprocessors have significantly higher performance and lower cost than specially-designed self-checked processors.

The most straightforward way to duplicate and compare a microprocessor is the technique of hard lockstepping, or running both microprocessors with the exact same clock. Each microprocessor is expected to produce the same outputs given the same inputs. Tandem Computers, IBM and Stratus Computers have all successfully used this technique. But hard lock-

stepped microprocessors are susceptible to minor non-determinisms and can have high failure rates due to otherwise correctable soft-errors. Furthermore, such a design can be unusable if a minor design error in the microprocessor renders the otherwise perfectly good part unusable for lock-stopped operation (this is a particularly troublesome problem, as a microprocessor vendor would be reluctant to undertake a costly design change for a problem that affects a relatively small market segment).

The Tandem Integrity S2 system moved away from the hard lockstepping approach and allowed each microprocessor to run on its own clock. To ensure deterministic operation, the microprocessor's performance counter is used to count retired instructions. Inputs to the microprocessors are carefully controlled. This technique allowed Tandem to use commodity microprocessors with only specially designed I/O interfaces.

The NSAA extends the duplicate and compare technique to allow each of the microprocessors to run more independently of the others. This allows running slices with different clock rates and even different microprocessor versions. Furthermore, NSAA allows asymmetric TLB fixup code and error fixup routines and even limited non-deterministic operation.

Current research in fault tolerant computing often discusses multiple threads doing serial execution of the same instruction stream [9]. This "replication in time" technique was first used in space-borne systems subject to high soft error rates. The technique is receiving renewed interest due to the high soft error rates in processors with sub 60nm features. These solutions have not been commercially viable in the fault tolerant data center for several reasons. Depending on how the checking is done, the approach might not be able to detect hard failures and can negatively impact performance. Furthermore, the checking covers processor execution units but not necessarily caches, memory controllers, and bus interconnects.

6. Conclusion

The NonStop Advanced Architecture avoids several roadblocks that would have affected new systems built with the prior NonStop architecture. The NSAA leverages existing 4-way Itanium2 servers, lowers development cost, and provides better customer value. And it does all this while increasing the fundamental deliverable of NonStop systems, better system availability for the customer's application.

The NonStop Advanced Architecture is the basis for all planned future NonStop systems. The MIPS®-based product line will be phased out in favor of the Itanium-based NSAA systems. New NSAA system designs are ongoing and NSAA will first ship to customers in mid-2005.

7. Acknowledgments

The authors thank Wendy Bartlett and David Schorow for their thoughtful reviews of earlier drafts of this paper. We also thank the anonymous DSN reviewers, whose perspective and useful comments helped us to improve the paper significantly. In addition, we gratefully acknowledge the many contributions of the hardware and software developers who are in the process of bringing this new architecture to life.

8. References

[1] "Commercial Fault Tolerance: A Tale of Two Systems", W. Bartlett and L. Spainhower, IEEE Transactions on Dependable and Secure Computing, Jan. – Mar. 2004.

[2] "ServerNet® II", D. Garcia, W. Watson, Proceedings of the Parallel Computer Routing and Communication Workshop, 1997

[3] "The Risk of Data Corruption in Microprocessor-based Systems", R. Horst, D. Jewett, and D. Lenoski, Proc 23rd International Symposium on Fault-Tolerant Computing, June 1993.

[4] "Reliable Computer Systems, Design and Evaluation", 2nd ed., pp 586-648, D. Siewiorek, R. Swarz, Digital Press, 1992.

[5] "InfiniBand® Architecture Specification Volume 1, Release 1.1", November 6, 2002. http://www.infinibandta.org/specs.

[6] "RDMA Protocol Specification", Version 1.0, R. Recio, P. Culley, D. Garcia, J. Hilland, http://www.rdmaconsortium.org/home/draft-recio-iwarp-rdmap-v1.0.pdf

[7] "Intel® Itanium® Architecture Software Developer's Manual", Intel Corporation, October 2002.

[8] "Reliable Computer Systems, Design and Evaluation", 2nd ed., pp 124-130, D. Siewiorek, R. Swarz, Digital Press, 1992.

[9] "Transient Fault Detection via Simultaneous Multithreading", S. Reinhardt, S. Mukherjee, Proceedings of the 27th Annual International Symposium on Computer Architecture, 2000.